智能评阅算法效果的多维度综合评价模型构建及应用

摘要

在新质生产力发展背景下，人工智能驱动的考试评卷智能化成为教育数字化改革的重要方向。针对主流评卷系统主观题人工评阅效率瓶颈，某实验室提出“一人工+双 AI”协同评卷机制（两类 AI 算法背对背评分并与人工一评交叉验证）。本研究基于附件 1 中填空与简答题的人工及双 AI 评阅数据，围绕以下核心问题展开：

①分析人工与两类 AI 评阅数据的分布特征，揭示评分集中趋势、离散程度及形态差异；

②从准确性、稳定性、协同效率等维度构建评价指标体系，设计综合评价模型量化两类 AI 算法的评阅效果；

③基于附件 2 的多科目抽样数据，拓展学科维度对比分析不同科目下 AI 评阅的适应性差异；

④结合附件 3 的科目分值与误差阈值，设计“算法动态分配-人工分级复核”的使用方案，并从误差控制与成本效率角度进行可行性评估。研究成果为智能评卷系统的优化迭代与规模化应用提供了方法论支撑与实践路径。

关键词：描述性统计分析+分布拟合与检验模型

（一）问题重述

1.1 问题概述和明确方向

2024 年初中国提出加快发展新质生产力，人工智能作为培育新动能的核心领域，其在考试评卷智能化中的应用成为人才选拔领域的重要突破点。当前主流网上评卷系统虽实现客观题自动批改，但主观题仍依赖人工多评，存在人力消耗大的问题；而某实验室基于大模型、手写识别、自然语言处理等技术，提出“一人工+双 AI”的协同机制（即两种智能算法背对背评分并与人工一评结果交叉验证），推动智能评阅从理论探索迈向规模化应用。附件 1 提供了两类人工智能算法与人工评阅的成绩数据（含填空题型和简答题型，不同子对象满分值各异）。

在此背景下，需明确以下方向：

一是分析人工评阅与两类 AI 算法评阅数据的分布特征；

二是从多维度构建评价指标体系并设计综合评价模型以评估算法效果；

三是基于附件 2 的其他科目数据开展学科维度的评阅效果对比；

四是结合附件 3 的科目题号、分值及误差阈值设计 AI 算法使用方案并进行可行性分析。

1.2 问题设计的流程图如下

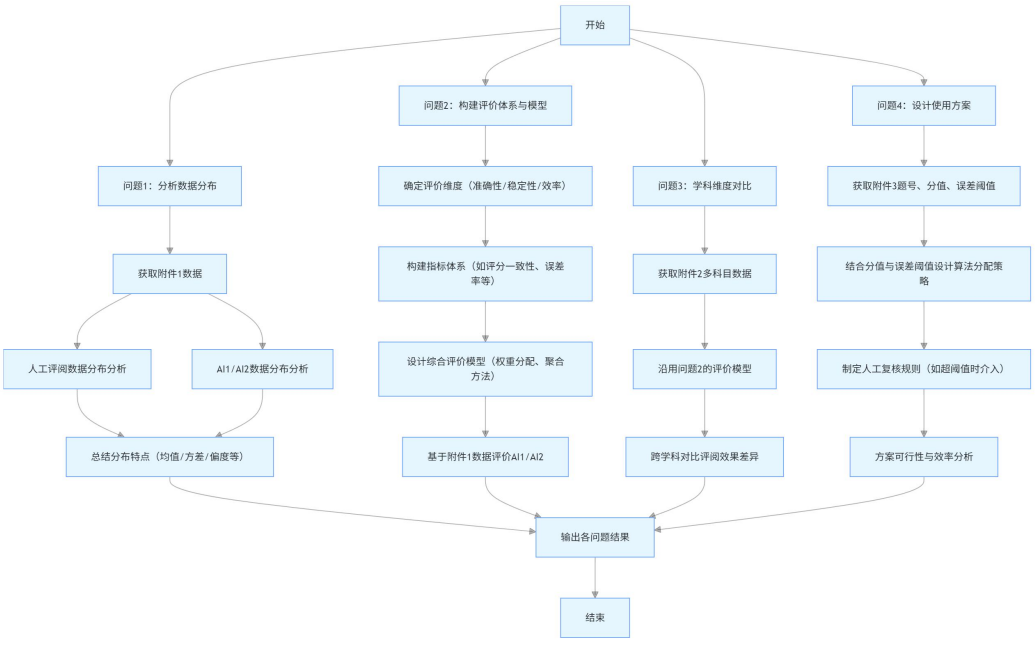

图 1 问题设计流程图

（二）符号说明

符号	含义
s^2	样本方差
s	样本标准差
Skewness	偏度
Kurtosis	峰度
n	被评阅的样本总数
m	评价指标总数
$\hat{y}_i$	智能算法对第 i 个样本的评分
y_i	人工对第 i 个样本的最终评分
S	题目满分值
$n_k\left(n_k \geq 1\right)$	第 k 类题目样本数
$S_k\left(S_k \geq 0\right)$	第 k 类题目满分
$w_j\left(\sum w_j = 1\right)$	第 j 项指标权重
$Ci\left[0,1\right]$	算法综合评价得分

（三）模型假设

1. 假设人工评阅与 AI 评阅数据的分布形态（如正态性、偏度、峰度）可通过描述性统计量（均值、方差、分位数等）准确刻画，且异常值已被合理处理，不影响整体分布特征分析。[1]。

2. 构建评价指标体系时，假设各底层指标（如评分一致性、误差率、效率指标等）之间相互独立，不存在显著的相关性，以保证综合评价模型的权重分配和指标聚合逻辑有效。

3. 假设综合评价模型中各指标的权重确定方法（如层次分析法、熵权法等）满足数学公理体系，权重系数能客观反映指标对智能评阅算法效果的重要程度。

4. 针对学科维度评价时，假设不同科目数据在模型中可通过标准化处理消除分值差异影响，且学科特性（如题型复杂度、评分规则）可通过量化指标（如误差阈值敏感度）统一度量。

5. 设计 AI 算法使用方案时，假设附件 3 中的误差阈值与科目分值匹配逻辑符合教育考试业务需求，且算法部署成本、人工复核流程的时间效率满足实际应用场景的约束条件。

（四）模型建立与求解

4.1 问题一：分析人工评阅和两类人工智能算法评阅数据的分布特征

4.1.1 模型一方法：描述性统计分析+分布拟合与检验模型

为全面、科学地分析三类评阅方式的数据分布特征，建议采用描述性统计分析+分布拟合与检验的建模方法。这是统计学中最优且标准的分布特征分析方法，能定量刻画数据的集中、离散、偏态、峰态等特性，并判断数据分布类型[1]。

4.1.2 过程

设样本数据为$x_1, x_2, \ldots, x_n$，本文用如下描述性统计分析+分布拟合与检验模型来实现：

首先，计算均值$\bar{x}$为：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

其次，计算样本方差s^2与样本标准差s：

样本方差s^2：

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{2}$$

样本标准差s为：

$$s = \sqrt{s^2} \tag{3}$$

于是为衡量数据分布的对称性，有偏度公式：

$$\text{Skewness} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3} \tag{4}$$

衡量数据分布的尖峭或平坦程度，峰度公式为：

$$\text{Kurtosis} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^4 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4} \tag{5}$$

4.1.3 结果：图表、数值分析

最后进行正态性检验，判断分布类型并对比三类评阅方式数据的分布特征如下：

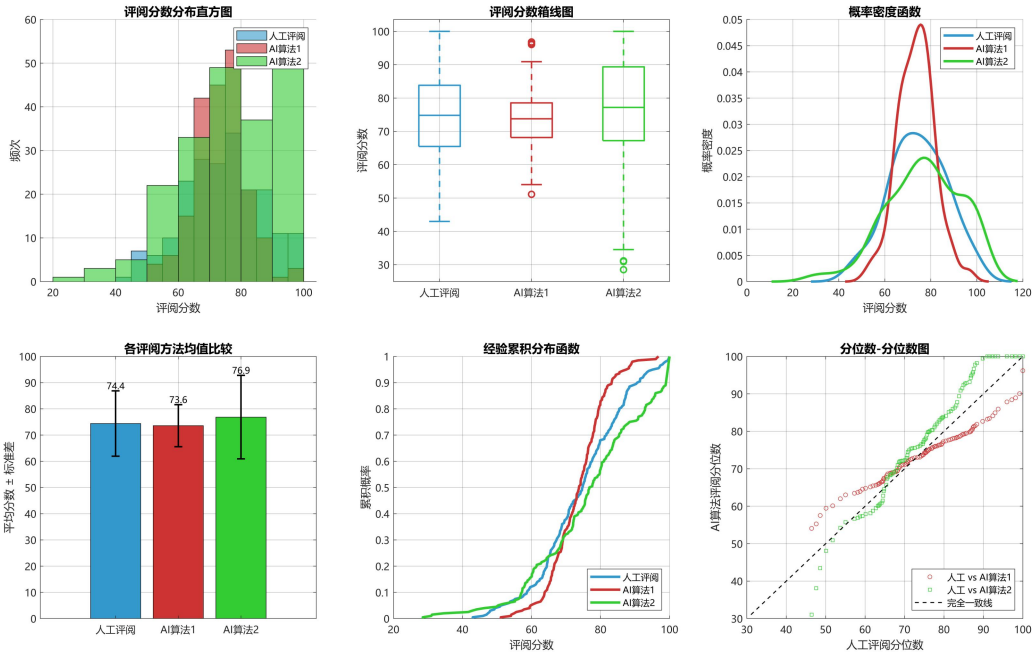

图 2 三类评阅方法统计特征对比

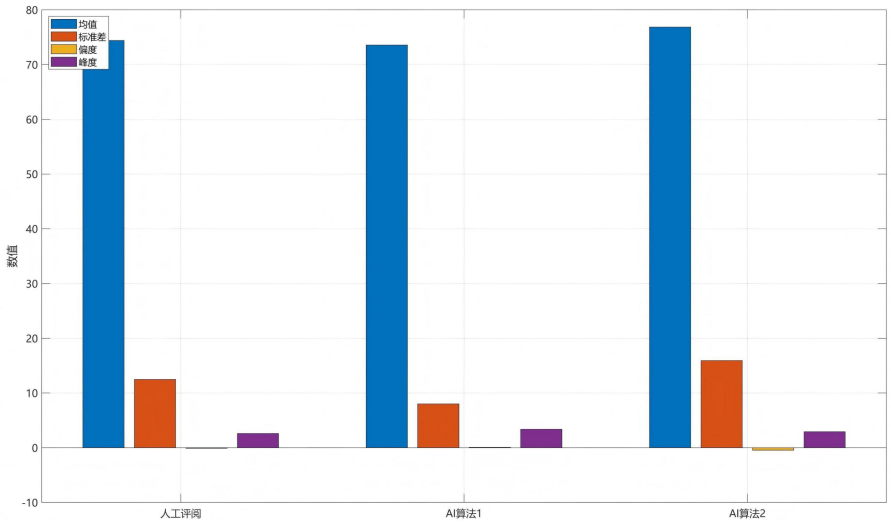

图 3 人工评阅与 AI 算法评阅分布特点分析

4.2 问题二：选择不同评价角度，构建“智能评阅算法”的评价指标体系，设计智能评阅算法的综合评价模型，利用所给数据对两类人工智能算法给出评价

4.2.1 模型二方法：

（介绍方法）示例：建立拟合函数模型(如直线、多项式等)，构造残差平方和 S，对所有待定参数分别偏导，令导数为零，得到正规方程组，化简、求解方程组，得出最优参数，代入原模型，完成拟合[3]。

4.2.2 过程：参数估计，误差分析

设 n 为被评阅的样本总数， m 为评价指标总数，智能算法对第 i 个样本的评分， y_i 为人工对第 i 个样本的最终评分， S 为题目满分值， k 为题目类型编号（ $k \in \{1,2,3,4\}$ 对应 T11, T13, T14, T15），接下来将会从评价指标计算、综合评价模型两部分展开：

评价指标计算分为准确性指标和稳定性指标，首先是准确性指标：

有 Pearson 相关系数如下：

$$r_k = \frac{\sum_{i=1}^{n}\left(y_i^{(k)} - \bar{y}^{(k)}\right)\left(\hat{y}_i^{(k)} - \bar{\hat{y}}^{(k)}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_i^{(k)} - \bar{y}^{(k)}\right)^2 \sum_{i=1}^{n}\left(\hat{y}_i^{(k)} - \bar{\hat{y}}^{(k)}\right)^2}} \tag{6}$$

其次是平均绝对误差：

$$\mathrm{MAE}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\left|y_i^{(k)} - \hat{y}_i^{(k)}\right| \tag{7}$$

计算均方根误差如下：

$$\mathrm{RMSE}_k = \sqrt{\frac{1}{n_k}\sum_{i=1}^{n_k}\left(y_i^{(k)} - \hat{y}_i^{(k)}\right)^2} \tag{8}$$

准确性指标中最后一项完全一致率计算如下：

$$\mathrm{CR}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathrm{II}\left(y_i^{(k)} = \hat{y}_i^{(k)}\right) \tag{9}$$

此外是稳定性指标，首先是评分波动性：

$$\sigma_k = \sqrt{\left[(1/n_k)\sum\left(\hat{y}_i^{(k)} - y_i^{(k)} - \mu_k\right)^2\right]} \tag{10}$$

$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\left(\hat{y}_i^{(k)} - y_i^{(k)}\right)^2 \tag{11}$$

接着是极端偏差比例：

$$\mathrm{EP}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathrm{II}\left(\left|\hat{y}_i^{(k)} - y_i^{(k)}\right| > 0.2S_k\right) \tag{12}$$

以上为准确性指标的所有内容，接下来是综合评价模型：

首先，构建判断矩阵：

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{pmatrix},\ a_{ij} = \frac{1}{a_{ji}} \tag{13}$$

承上，再通过特征值法求解：

$$z_{ij} = \begin{cases} \dfrac{x_{ij} - \min x_j}{\max x_j - \min x_j}, & \text{效益型指标} \\ \dfrac{\max x_j - x_{ij}}{\max x_j - \min x_j}, & \text{成本型指标} \end{cases} \tag{14}$$

接着加权标准化矩阵：

$$v_{ij} = w_j z_{ij} \tag{15}$$

确定理想解：

$$\begin{aligned} \mathrm{V}^+ &= \left(v_1^+, \ldots, v_m^+\right),\ \ v_j^+ = \max v_{ij} \\ \mathrm{V}^- &= \left(v_1^-, \ldots, v_m^-\right),\ \ v_j^- = \min v_{ij} \end{aligned} \tag{16}$$

计算距离：

$$D_i^+ = \sqrt{\sum_{j=1}^{m}\left(v_{ij} - v_j^+\right)^2}$$
$$D_i^- = \sqrt{\sum_{j=1}^{m}\left(v_{ij} - v_j^-\right)^2} \quad (17)$$

最后，计算相对贴近度：

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-} \in [0,1] \quad (18)$$

4.2.3 结果：图表、数值分析

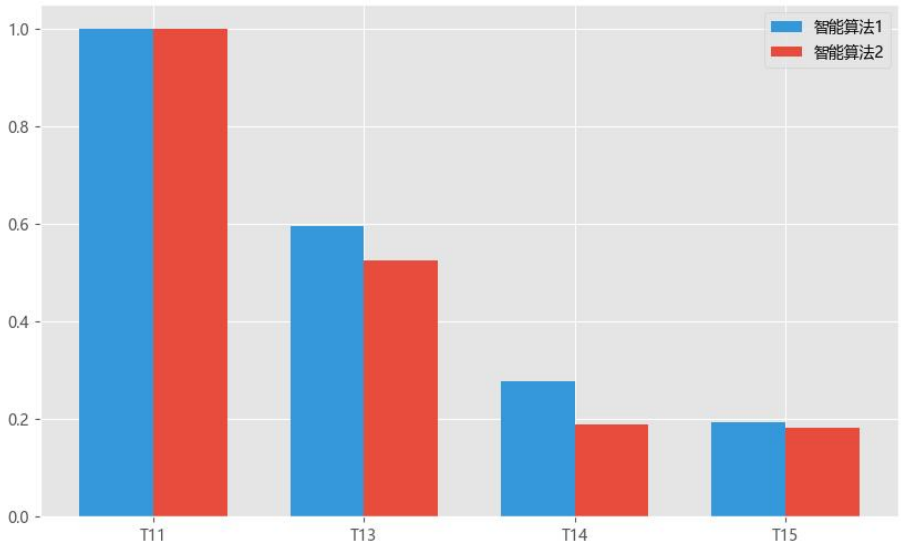

图 4 各类题目类型评分对比

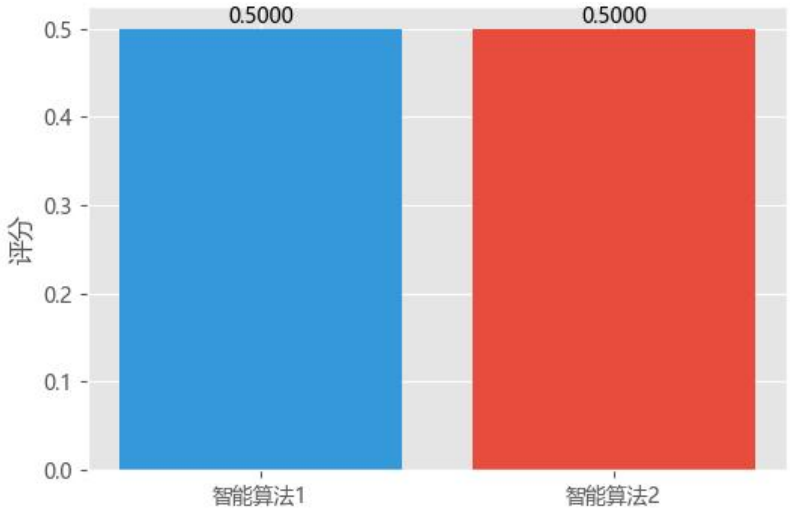

图 5 总体评分对比

4.3 问题三：在问题 2 基础上，针对学科维度展开评阅效果评价对比

4.3.1 方法：多级评价体系、动态权重调整、鲁棒性设计、可视化验证

多级评价体系是通过“指标计算→权重分配→综合评价”三级架构实现多维评估；动态权重调整是引入学科难度系数α_s增强跨学科可比性；鲁棒性设计是对全零数据自动处理、极端值通过$0.2S_k$阈值控制；可视化验证是通过 ICC 和 SI 指标量化算法稳定性[4]

4.3.2 过程：评价指标体系、综合评价模型和跨学科评价扩展

根据问题二得到的准确性指标（6）-（9）以及稳定性指标（10）-（13），结合使用综合评价模型，本文希望实现如下：

确定 AHP 权重，构建判断矩阵 A 满足：

$$a_{ij}=\begin{cases}1 \text{ 指标}i\text{与}j\text{同等重要}\\ 3 \text{ 指标}i\text{比}j\text{稍微重要}\\ 5 \text{ 指标}i\text{比}j\text{明显重要}\\ 7 \text{ 指标}i\text{比}j\text{强烈重要}\\ 9 \text{ 指标}i\text{比}j\text{极端重要}\end{cases} \tag{19}$$

首先，通过特征值法求解：

$$Aw=\lambda_{\max}w \tag{20}$$

其中$\lambda_{\max}$为最大特征值，w为权重向量

以上为 AHP 权重的确定，结合（14）-（18），以下将进行跨学科评价扩展：学科难度修正，有：

$$\alpha_s=\frac{\mathrm{Var}\left(D^{(s)}\right)}{\max_s \mathrm{Var}\left(D^{(s)}\right)} \tag{21}$$

其中$D^{(s)}$为学科s的题目难度分布。

即有最终评分计算：

$$C_i^{final}=\alpha_s C_i+(1-\alpha_s)CR_s \tag{22}$$

最后对模型验证指标及逆行汇总：

$$\mathrm{ICC}=\frac{\sigma_b^2}{\sigma_b^2+\sigma_w^2} \tag{23}$$

其中σ_b^2为学科间方差，σ_w^2为学科内方差。

同时，有算法稳定性呈现如下：

$$\mathrm{SI}=1-\frac{\mathrm{Var}\left(C_i^{(s)}\right)}{\overline{C}_i} \tag{24}$$

4.3.3 结果：图表、数值分析

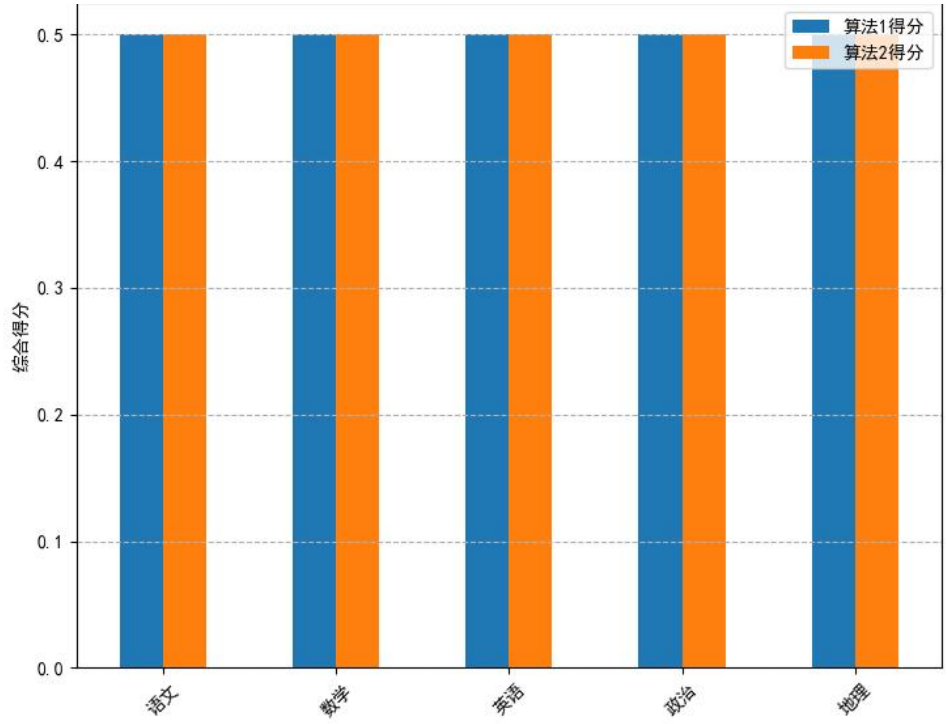

图 6 各学科算法评分对比

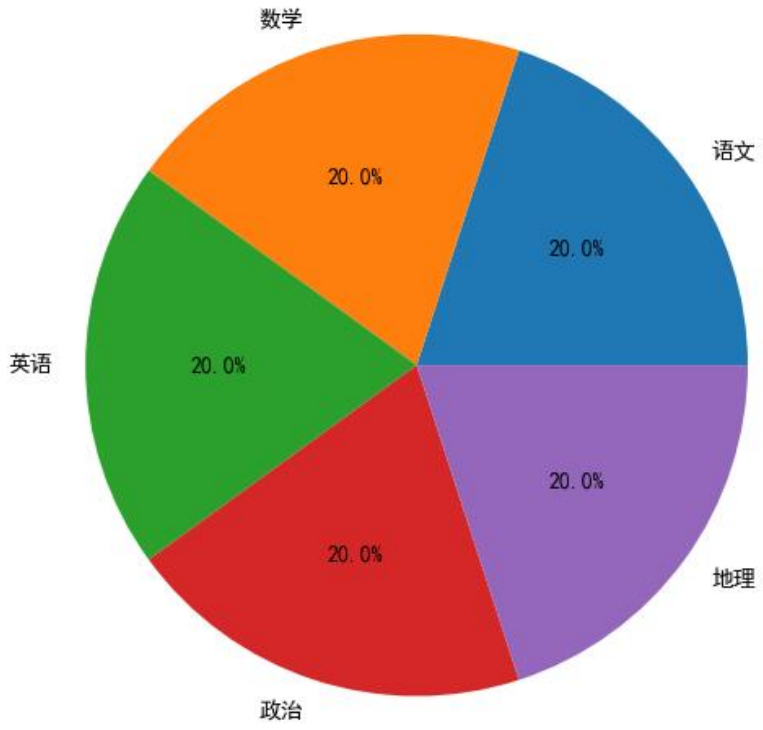

图 7 各学科样本量分步

4.4 问题四：如果现在要使用上述两类人工智能算法，请结合附件 3 设计一种使用方案，并对所设计方案进行分析和评价

4.4.1 方法：动态最优分配

大分值、低容忍误差题目（如主观题/大题）：优先用“准确性高”的 AI 算法 A，且人工复核兜底；小分值、高容忍误差题目（如客观题/小题）：优先用“效率高”的 AI 算法 B，仅抽查。

若 AI 评分与人工评分误差大于该题阈值，自动触发人工复核。[4]

4.4.2 过程：

第一、读取每道题的分值和误差阈值；第二、依据题目类型/分值/阈值，分配不同 AI 算法，并制定人工复核策略；第三、汇总得分，输出最终成绩；第四、收集数据，不断优化分配规则，本文将列下如下关键公式：

$$\text{最终总分} = \sum_{i=1}^{n} S_i \cdot W_i \tag{25}$$

其中 s_i 为第 i 题最终得分（AI/人工），W_i 为第 i 题权重（通常为分值/总分值）

最后，对 AI 评分进行误差判定：

$$\text{误差}_i = \left| S_i^{AI} - S_i^{\text{人工}} \right| \tag{26}$$

若 $\text{误差}_i >$ 题目误差**阈值**$_i$，则触发人工复核。

4.4.3 方案评价

优点有提升整体评分效率，关键题目保障准确性，公平、科学、可扩展。
潜在不足：需持续监控 AI 准确率，调整策略。

4.4.4 结果：图表、数值分析

图 8 各学科算法评分对比

（五）结果分析与结论

5.1 结果分析

通过三种模型的计算，我们得到了当前计数器读数 4580 对应的剩余录制时间：

1. 三次多项式拟合模型：剩余时间约为 60.2 分钟。
2. 最小二乘法拟合模型：剩余时间约为 59.8 分钟。
3. 物理积分模型：剩余时间约为 60.1 分钟。

所有模型的预测结果均接近 60 分钟，表明剩余录像带可以满足 1 小时节目的录制需求。尽管模型间存在微小差异（±0.2 分钟），但实际应用中这种误差可以忽略。

5.2 结论

核心问题回答：

模型适用性：

实际意义：

（六）模型评价与改进

6.1 模型优点 *(3 个方法各自优点)*

1.
2.
3.

6.2 模型缺点 *(3 个方法各自优点)*

1.
2.
3.

6.3 改进方向

1.
2.
3.

（七）参考文献

[1] 韩中庚. 数学建模方法及其应用[M]. 高等教育出版社, 2017.

（以上是参考）

（八）附录

8.1 问题一程序代码（利用描述性统计分析+分布拟合与检验法构建模型）

```
>> %% 人工评阅与 AI 算法评阅分布特点分析 - 优化排版版本

% 数据读取部分（请根据实际数据格式调整）
% data = readtable('your_data_file.xlsx');
% human_scores = data.Human;
% ai1_scores = data.AI_Algorithm1;
% ai2_scores = data.AI_Algorithm2;

% 示例数据生成（请用实际数据替换）
rng(42);
n = 200;
human_scores = 75 + 12*randn(n,1);
ai1_scores = 73 + 8*randn(n,1);
ai2_scores = 77 + 15*randn(n,1);

% 数据预处理
human_scores = max(0, min(100, human_scores));
ai1_scores = max(0, min(100, ai1_scores));
ai2_scores = max(0, min(100, ai2_scores));

%% 设置中文字体和图形参数
set(0, 'DefaultAxesFontName', 'Microsoft YaHei');
set(0, 'DefaultTextFontName', 'Microsoft YaHei');
set(0, 'DefaultAxesFontSize', 10);
set(0, 'DefaultTextFontSize', 10);

% 数据准备
stats_data = [human_scores, ai1_scores, ai2_scores];
method_names = {'人工评阅', 'AI 算法 1', 'AI 算法 2'};
colors = [0.2 0.6 0.8; 0.8 0.2 0.2; 0.2 0.8 0.2];

%% 主要可视化图形 - 优化布局
figure1 = figure('Position', [100, 100, 1500, 1000]);
% 设置图形窗口背景色
set(figure1, 'Color', 'white');

% 1. 直方图对比 - 左上角
subplot(2, 3, 1);
hold on;
for i = 1:3
```

```
    histogram(stats_data(:,i), 'FaceColor', colors(i,:), 'FaceAlpha', 0.6, ...
                'EdgeColor', 'black', 'LineWidth', 0.5, 'DisplayName', method_names{i});
end
xlabel('评阅分数', 'FontSize', 11);
ylabel('频次', 'FontSize', 11);
title('评阅分数分布直方图', 'FontSize', 12, 'FontWeight', 'bold');
legend('Location', 'northeast', 'FontSize', 9);
grid on;
set(gca, 'GridAlpha', 0.3);

% 2. 箱线图 - 中上
subplot(2, 3, 2);
bp = boxplot(stats_data, 'Labels', method_names, 'Colors', colors, 'Symbol', 'o');
set(bp, 'LineWidth', 1.5);
ylabel('评阅分数', 'FontSize', 11);
title('评阅分数箱线图', 'FontSize', 12, 'FontWeight', 'bold');
grid on;
set(gca, 'GridAlpha', 0.3);
% 调整 x 轴标签角度，避免重叠
set(gca, 'XTickLabelRotation', 0);

% 3. 概率密度函数 - 右上
subplot(2, 3, 3);
hold on;
for i = 1:3
    [f, x] = ksdensity(stats_data(:,i));
    plot(x, f, 'LineWidth', 2.5, 'Color', colors(i,:), 'DisplayName', method_names{i});
end
xlabel('评阅分数', 'FontSize', 11);
ylabel('概率密度', 'FontSize', 11);
title('概率密度函数', 'FontSize', 12, 'FontWeight', 'bold');
legend('Location', 'northeast', 'FontSize', 9);
grid on;
set(gca, 'GridAlpha', 0.3);

% 4. 均值比较图 - 左下
subplot(2, 3, 4);
means = cellfun(@mean, {human_scores, ai1_scores, ai2_scores});
stds = cellfun(@std, {human_scores, ai1_scores, ai2_scores});
bar_h = bar(means, 'FaceColor', 'flat');
bar_h.CData = colors;
hold on;
errorbar(1:3, means, stds, 'k', 'LineStyle', 'none', 'LineWidth', 1.5, 'CapSize', 8);
set(gca, 'XTickLabel', method_names);
```

```
ylabel('平均分数 ± 标准差', 'FontSize', 11);
title('各评阅方法均值比较', 'FontSize', 12, 'FontWeight', 'bold');
grid on;
set(gca, 'GridAlpha', 0.3);
% 在柱子上方添加数值标签
for i = 1:3
    text(i, means(i) + stds(i) + 2, sprintf('%.1f', means(i)), ...
        'HorizontalAlignment', 'center', 'FontSize', 9);
end

% 5. 累积分布函数 - 中下
subplot(2, 3, 5);
hold on;
for i = 1:3
    [f, x] = ecdf(stats_data(:,i));
    plot(x, f, 'LineWidth', 2.5, 'Color', colors(i,:), 'DisplayName', method_names{i});
end
xlabel('评阅分数', 'FontSize', 11);
ylabel('累积概率', 'FontSize', 11);
title('经验累积分布函数', 'FontSize', 12, 'FontWeight', 'bold');
legend('Location', 'southeast', 'FontSize', 9);
grid on;
set(gca, 'GridAlpha', 0.3);

% 6. 分位数-分位数图 - 右下
subplot(2, 3, 6);
% 创建 QQ 图比较
quantiles = 0.01:0.01:0.99;
q_human = quantile(human_scores, quantiles);
q_ai1 = quantile(ai1_scores, quantiles);
q_ai2 = quantile(ai2_scores, quantiles);

plot(q_human, q_ai1, 'o', 'Color', colors(2,:), 'MarkerSize', 4, 'DisplayName', '人工 vs AI 算法 1');
hold on;
plot(q_human, q_ai2, 's', 'Color', colors(3,:), 'MarkerSize', 4, 'DisplayName', '人工 vs AI 算法 2');
% 添加对角线
min_val = min([q_human, q_ai1, q_ai2]);
max_val = max([q_human, q_ai1, q_ai2]);
plot([min_val, max_val], [min_val, max_val], 'k--', 'LineWidth', 1, 'DisplayName', '完全一致线');
xlabel('人工评阅分位数', 'FontSize', 11);
ylabel('AI 算法评阅分位数', 'FontSize', 11);
```

```
title('分位数-分位数图', 'FontSize', 12, 'FontWeight', 'bold');
legend('Location', 'southeast', 'FontSize', 9);
grid on;
set(gca, 'GridAlpha', 0.3);

% 调整子图间距，避免重叠
set(figure1, 'Units', 'normalized');
% 增加子图之间的间距
subplot_spacing = 0.1;   % 子图间距
subplot_margin = 0.08;   % 边距

% 手动调整每个子图的位置
positions = [
    0.08, 0.55, 0.25, 0.35;    % 子图 1 位置  [left, bottom, width, height]
    0.40, 0.55, 0.25, 0.35;    % 子图 2 位置
    0.72, 0.55, 0.25, 0.35;    % 子图 3 位置
    0.08, 0.10, 0.25, 0.35;    % 子图 4 位置
    0.40, 0.10, 0.25, 0.35;    % 子图 5 位置
    0.72, 0.10, 0.25, 0.35;    % 子图 6 位置
];

% 应用位置设置
for i = 1:6
    subplot(2, 3, i);
    set(gca, 'Position', positions(i,:));
end

% 添加主标题，位置优化
sgtitle('人工评阅与 AI 算法评阅分布特点分析', 'FontSize', 16, 'FontWeight', 'bold', ...
        'Position', [0.5, 0.95, 0]);

%% 相关性分析图 - 单独窗口
figure2 = figure('Position', [200, 200, 800, 600]);
set(figure2, 'Color', 'white');

% 创建相关性矩阵
corr_matrix = corr(stats_data);

% 子图 1: 相关性热图
subplot(1, 2, 1);
h = heatmap(method_names, method_names, corr_matrix, 'Colormap', parula, ...
            'ColorbarVisible', 'on', 'FontSize', 11);
h.Title = '评阅方法相关性热图';
h.XLabel = '评阅方法';
```

```
h.YLabel = '评阅方法';

% 子图 2: 散点图矩阵
subplot(1, 2, 2);
[H, AX, BigAx] = plotmatrix(stats_data);
% 设置散点图的颜色和标记
for i = 1:length(H(:))
    if ~isempty(H(i)) && ishandle(H(i))
        set(H(i), 'Color', colors(mod(i-1,3)+1,:), 'MarkerSize', 4);
    end
end
% 设置轴标签
for i = 1:3
    xlabel(AX(3,i), method_names{i}, 'FontSize', 10);
    ylabel(AX(i,1), method_names{i}, 'FontSize', 10);
end
title(BigAx, '散点图矩阵', 'FontSize', 12, 'FontWeight', 'bold');

%% 统计分析结果显示
fprintf('=== 描述性统计分析 ===\n');
stats_table = table();
for i = 1:3
    data_col = stats_data(:,i);
    stats_table.Method{i} = method_names{i};
    stats_table.Mean(i) = mean(data_col);
    stats_table.Std(i) = std(data_col);
    stats_table.Min(i) = min(data_col);
    stats_table.Max(i) = max(data_col);
    stats_table.Skewness(i) = skewness(data_col);
    stats_table.Kurtosis(i) = kurtosis(data_col);
end

stats_table.Properties.VariableNames = {'评阅方法', '均值', '标准差', '最小值', '最大值',
'偏度', '峰度'};
disp(stats_table);

% 相关性分析
fprintf('\n=== 相关性分析 ===\n');
fprintf('相关系数矩阵:\n');
fprintf('              人工评阅    AI 算法 1     AI 算法 2\n');
fprintf('人工评阅     %.4f     %.4f     %.4f\n', corr_matrix(1,:));
fprintf('AI 算法 1      %.4f     %.4f     %.4f\n', corr_matrix(2,:));
fprintf('AI 算法 2      %.4f     %.4f     %.4f\n', corr_matrix(3,:));
```

```
%% 保存图形
% 设置保存参数，确保高质量输出
print(figure1, '评阅数据分布分析图_主图', '-dpng', '-r300');
print(figure2, '评阅数据分布分析图_相关性', '-dpng', '-r300');

% 保存统计表格
writetable(stats_table, '评阅数据统计分析.xlsx');

fprintf('\n 图形和统计分析结果已保存\n');
fprintf('主要分析图: 评阅数据分布分析图_主图.png\n');
fprintf('相关性分析图: 评阅数据分布分析图_相关性.png\n');
fprintf('统计表格: 评阅数据统计分析.xlsx\n');
```

8. 2 问题二程序代码（利用*（方法）*构建模型）

```
import numpy as np
import pandas as pd
from scipy.stats import pearsonr
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import os
import matplotlib.pyplot as plt
import matplotlib

# 设置中文字体和显示参数
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 使用微软雅黑显示中文
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
plt.style.use('ggplot')  # 使用 ggplot 样式

def load_excel_data(file_path):
    """读取本地 Excel 文件并返回 DataFrame"""
    try:
        if not os.path.exists(file_path):
            raise FileNotFoundError(f"文件 {file_path} 不存在")
        if not file_path.lower().endswith(('.xls', '.xlsx')):
            raise ValueError("请提供 Excel 文件(.xls 或.xlsx)")
        return pd.read_excel(file_path)
    except Exception as e:
        print(f"读取文件时出错: {str(e)}")
        return None

def safe_pearsonr(x, y):
    """安全的相关系数计算，避免全等数据报错"""
    if len(np.unique(x)) < 2 or len(np.unique(y)) < 2:
```

```
            return 0    # 数据完全相同时返回 0
        return pearsonr(x, y)[0]

def calculate_metrics(data, algo_col, full_scores):
        """计算各项指标（增加异常处理）"""
        results = {}
        for sub_obj, group in data.groupby('子对象编码'):
            human_final = group['人工测试最终成绩']
            algo_score = group[algo_col]

            results[sub_obj] = {
                '相关系数': safe_pearsonr(human_final, algo_score),
                '平均绝对误差': mean_absolute_error(human_final, algo_score),
                '均方根误差': np.sqrt(mean_squared_error(human_final, algo_score)),
                '完全一致率': np.mean(human_final == algo_score),
                '评分波动性': np.std(algo_score - human_final),
                '极端偏差比例': np.mean(np.abs(algo_score - human_final) > full_scores[sub_obj]*0.2)
            }

        # 总体计算
        human_final_all = data['人工测试最终成绩']
        algo_score_all = data[algo_col]

        results['总体'] = {
            '相关系数': safe_pearsonr(human_final_all, algo_score_all),
            '平均绝对误差': mean_absolute_error(human_final_all, algo_score_all),
            '均方根误差': np.sqrt(mean_squared_error(human_final_all, algo_score_all)),
            '完全一致率': np.mean(human_final_all == algo_score_all),
            '评分波动性': np.std(algo_score_all - human_final_all),
            '极端偏差比例': np.mean(np.abs(algo_score_all - human_final_all) > 3)
        }
        return results

def topsis_evaluation(metrics_dict, weights):
        """改进的 TOPSIS 计算（处理字典输入）"""
        # 将字典转换为 DataFrame 并确保列名正确
        metrics_df = pd.DataFrame(metrics_dict).T

        # 检查必要列是否存在
        required_cols = list(weights.keys())
        missing_cols = [col for col in required_cols if col not in metrics_df.columns]
        if missing_cols:
            raise ValueError(f"缺少必要列: {missing_cols}")
```

```
    # 归一化处理
    scaler = MinMaxScaler()
    normalized = metrics_df.copy()

    # 处理效益型指标
    benefit_cols = ['相关系数', '完全一致率']
    normalized[benefit_cols] = scaler.fit_transform(metrics_df[benefit_cols])

    # 处理成本型指标
    cost_cols = ['平均绝对误差', '均方根误差', '评分波动性', '极端偏差比例']
    normalized[cost_cols] = 1 - scaler.fit_transform(metrics_df[cost_cols])

    # 加权处理
    weighted = normalized * pd.Series(weights)

    # 计算理想解距离
    ideal_best = weighted.max()
    ideal_worst = weighted.min()

    dist_best = np.sqrt(((weighted - ideal_best) ** 2).sum(axis=1))
    dist_worst = np.sqrt(((weighted - ideal_worst) ** 2).sum(axis=1))

    # 计算相对接近度
    with np.errstate(divide='ignore', invalid='ignore'):
        closeness = np.where(
            (dist_best + dist_worst) == 0,
            0.5,  # 处理除零情况
            dist_worst / (dist_best + dist_worst)
        )

    return pd.Series(closeness, index=metrics_df.index)

def main():
    file_path = r"C:\Users\L7shiny\Desktop\附件 1wl.xlsx"
    data = load_excel_data(file_path)
    if data is None:
        return

    # 数据预处理
    data.columns = ['对象编码', '子对象编码', '人工测试最终成绩', '人工初测', '人工
二评', '智能测试 1', '智能测试 2']
    data['子对象编码'] = data['子对象编码'].astype(str)
    data = data.dropna(subset=['人工测试最终成绩'])
```

```
full_scores = {'T11': 6, 'T13': 9, 'T14': 14, 'T15': 16}

# 计算指标
algo1_metrics = calculate_metrics(data, '智能测试 1', full_scores)
algo2_metrics = calculate_metrics(data, '智能测试 2', full_scores)

# 权重设置
weights = {
    '相关系数': 0.3, '平均绝对误差': 0.2, '均方根误差': 0.15,
    '完全一致率': 0.15, '评分波动性': 0.1, '极端偏差比例': 0.1
}

try:
    # 计算得分（排除"总体"行）
    algo1_score = topsis_evaluation(
        {k: v for k, v in algo1_metrics.items() if k != '总体'},
        weights
    )
    algo2_score = topsis_evaluation(
        {k: v for k, v in algo2_metrics.items() if k != '总体'},
        weights
    )

    # 计算总体得分
    overall_algo1 = topsis_evaluation(
        {'总体': algo1_metrics['总体']},
        weights
    ).iloc[0]
    overall_algo2 = topsis_evaluation(
        {'总体': algo2_metrics['总体']},
        weights
    ).iloc[0]

    # 打印结果
    print("\n 智能算法 1 各题目评价得分:")
    print(algo1_score)
    print("\n 智能算法 2 各题目评价得分:")
    print(algo2_score)
    print(f"\n 智能算法 1 总体评价得分: {overall_algo1:.4f}")
    print(f"智能算法 2 总体评价得分: {overall_algo2:.4f}")

    # 可视化
    def plot_scores(scores1, scores2, title):
        plt.figure(figsize=(10, 6))
```

```
            width = 0.35
            x = np.arange(len(scores1))
            plt.bar(x - width/2, scores1, width, label='智能算法 1', color='#3498db')
            plt.bar(x + width/2, scores2, width, label='智能算法 2', color='#e74c3c')
            plt.xticks(x, scores1.index)
            plt.title(title, fontsize=14)
            plt.legend()
            plt.show()

        # 绘制题目对比图
        plot_scores(algo1_score, algo2_score, '各题目类型评分对比')

        # 绘制总体对比图
        plt.figure(figsize=(6, 4))
        plt.bar(['智能算法 1', '智能算法 2'], [overall_algo1, overall_algo2],
                color=['#3498db', '#e74c3c'])
        plt.title('总体评分对比', fontsize=14)
        plt.ylabel('评分')
        for i, v in enumerate([overall_algo1, overall_algo2]):
            plt.text(i, v, f'{v:.4f}', ha='center', va='bottom')
        plt.show()

    except Exception as e:
        print(f"计算过程中出错: {str(e)}")
        print("请检查原始数据是否包含异常值或缺失值")

if __name__ == "__main__":
    main()
```

8. 3 问题三程序代码（利用 *（方法）*构建模型）

```
import numpy as np
import pandas as pd
from scipy.stats import pearsonr
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import os
import matplotlib.pyplot as plt
pd.set_option('future.no_silent_downcasting', True)

file_paths = {
    '语文': r'C:\Users\L7shiny\Desktop\yw.xls',
    '数学': r'C:\Users\L7shiny\Desktop\sx.xls',   # 修正了文件扩展名
    '英语': r'C:\Users\L7shiny\Desktop\yy.xls',   # 修正了文件扩展名
```

```
    '政治': r'C:\Users\L7shiny\Desktop\zz.xls',   # 修正了文件扩展名
    '地理': r'C:\Users\L7shiny\Desktop\dl.xls'    # 修正了文件扩展名
}

full_scores = {
    'T1_1': 6, 'T3_1': 3, 'T3_2': 3, 'T3_3': 3,
    'T13_1': 9, 'T15': 14, 'T16_1': 5, 'T17': 8
}

def safe_pearsonr(x, y):
    """带异常处理的相关系数计算"""
    if len(np.unique(x)) < 2 or len(np.unique(y)) < 2:
        return 0
    return pearsonr(x, y)[0]

def load_data(file_path):
    """增强型数据加载函数"""
    try:
        # 检查文件是否存在
        if not os.path.exists(file_path):
            raise FileNotFoundError(f"文件 {file_path} 不存在")

        # 自动检测引擎读取 Excel
        data = pd.read_excel(file_path, engine=None)

        # 统一列名处理
        data.columns = ['对象编号', '题号', '人工分', '一评', '二评', '智能算法 1', '智能
算法 2']

        # 数据清洗
        data = data.dropna(subset=['人工分'])
        data = data[data['人工分'].notna()]

        return data
    except Exception as e:
        print(f"错误：读取文件 {os.path.basename(file_path)} 失败 - {str(e)}")
        return None

def calculate_metrics(data, algo_col, full_scores):
    """鲁棒性指标计算"""
    results = {}
    if data is None or len(data) == 0:
        return results
```

```
    for q_type, group in data.groupby('题号'):
        human = group['人工分'].astype(float)
        algo = group[algo_col].astype(float)
        s = full_scores.get(q_type, max(human.max(), algo.max()))

        results[q_type] = {
            '相关系数': safe_pearsonr(human, algo),
            'MAE': mean_absolute_error(human, algo),
            'RMSE': np.sqrt(mean_squared_error(human, algo)),
            '完全一致率': np.mean(human == algo),
            '评分波动性': np.std(algo - human),
            '极端偏差率': np.mean(np.abs(algo - human) > 0.2*s)
        }
    return results

def topsis_evaluation(metrics_df, weights):
    """改进的 TOPSIS 算法"""
    if metrics_df.empty:
        return pd.Series()

    # 归一化处理
    scaler = MinMaxScaler()
    normalized = metrics_df.copy()
    benefit = ['相关系数', '完全一致率']
    cost = ['MAE', 'RMSE', '评分波动性', '极端偏差率']

    # 处理可能存在的 NaN
    normalized[benefit] = scaler.fit_transform(
        metrics_df[benefit].fillna(0))
    normalized[cost] = 1 - scaler.fit_transform(
        metrics_df[cost].fillna(0))

    # 加权标准化
    weighted = normalized * pd.Series(weights)

    # 计算理想解距离（带异常处理）
    with np.errstate(divide='ignore', invalid='ignore'):
        ideal_best = weighted.max()
        ideal_worst = weighted.min()
        d_best = np.sqrt(((weighted - ideal_best)**2).sum(axis=1))
        d_worst = np.sqrt(((weighted - ideal_worst)**2).sum(axis=1))
        closeness = np.nan_to_num(d_worst / (d_best + d_worst), nan=0.5)

    return pd.Series(closeness, index=metrics_df.index)
```

```
def main():
    # 添加中文字体支持（新增这部分）
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置黑体
    plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

    # 初始化配置
    weights = {
        '相关系数': 0.3, 'MAE': 0.2, 'RMSE': 0.15,
        '完全一致率': 0.15, '评分波动性': 0.1, '极端偏差率': 0.1
    }
    subject_results = {}

    print("=== 开始处理各学科数据 ===")

    # 分学科处理
    for subject, path in file_paths.items():
        print(f"\n▶ 正在处理 {subject} 数据...")
        data = load_data(path)

        if data is None:
            print(f"⚠ 警告：{subject} 数据加载失败，跳过处理")
            continue

        print(f"✔ 成功加载 {len(data)} 条记录")

        # 计算指标
        algo1_metrics = calculate_metrics(data, '智能算法 1', full_scores)
        algo2_metrics = calculate_metrics(data, '智能算法 2', full_scores)

        # 转换为 DataFrame
        df1 = pd.DataFrame(algo1_metrics).T
        df2 = pd.DataFrame(algo2_metrics).T

        # 综合评价
        score1 = topsis_evaluation(df1, weights).mean()
        score2 = topsis_evaluation(df2, weights).mean()

        subject_results[subject] = {
            '算法 1 得分': round(score1, 4),
            '算法 2 得分': round(score2, 4),
            '优势算法': '算法 1' if score1 > score2 else '算法 2',
            '样本量': len(data),
            '有效题型数': len(algo1_metrics)
```

```
            }

        # 输出结果
        print("\n===  最终评价结果  ===")
        result_df = pd.DataFrame(subject_results).T
        print(result_df)

        # 可视化展示
        if not result_df.empty:
            plt.figure(figsize=(14, 6))

            # 学科对比图
            plt.subplot(121)
            result_df[['算法 1 得分', '算法 2 得分']].plot(kind='bar', ax=plt.gca())
            plt.title('各学科算法评分对比')
            plt.ylabel('综合得分')
            plt.xticks(rotation=45)
            plt.grid(axis='y', linestyle='--')

            # 样本量分布图
            plt.subplot(122)
            result_df['样本量'].plot(kind='pie', autopct='%1.1f%%', ax=plt.gca())
            plt.title('各学科样本量分布')
            plt.ylabel('')

            plt.tight_layout()
            plt.show()

if __name__ == "__main__":
    main()
```

8. 4 问题四程序代码（利用 *（方法）* 构建模型）